let's move
the java world

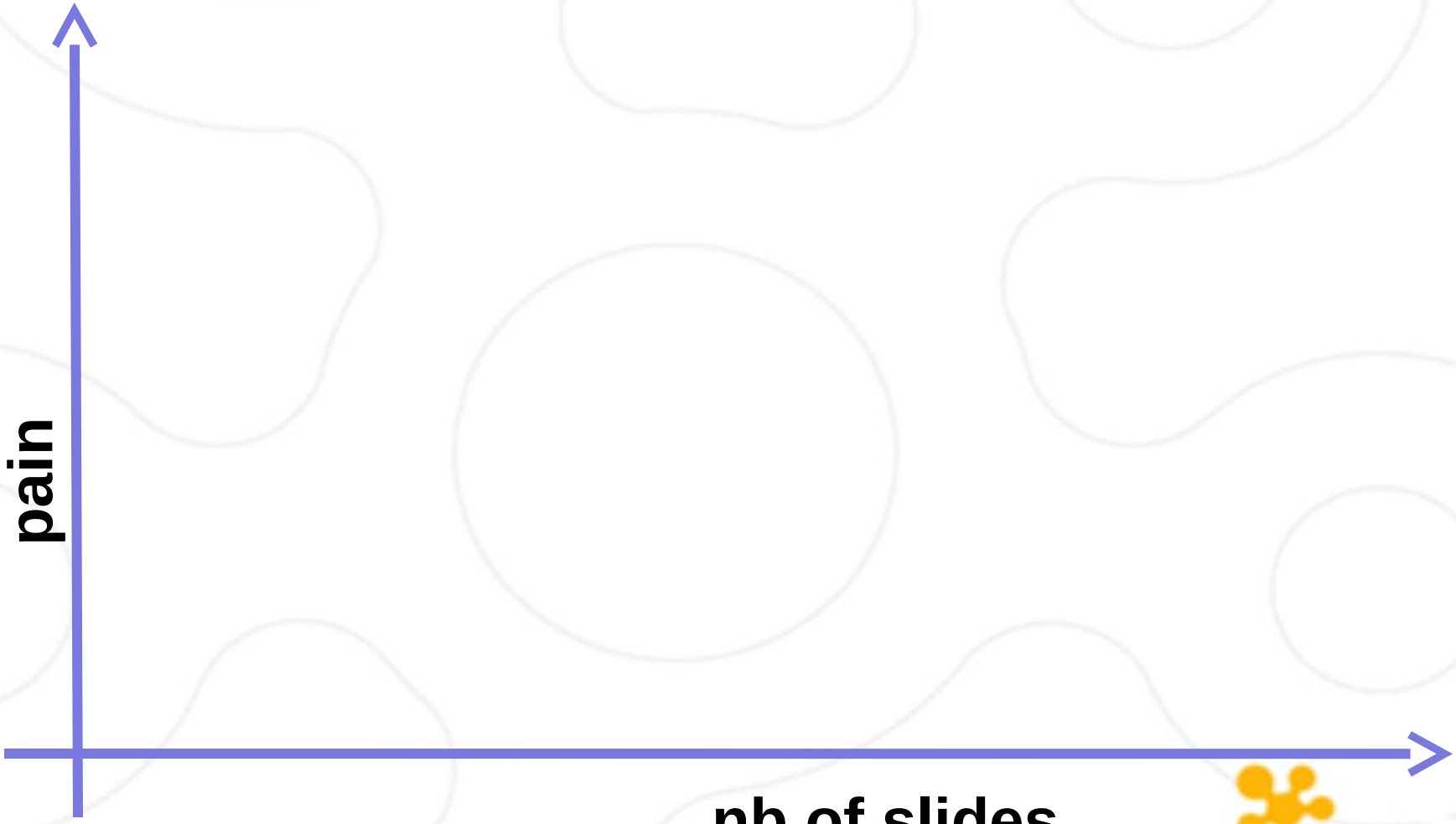
EJB applications guided by tests

Jakub Marchwicki

The experiment

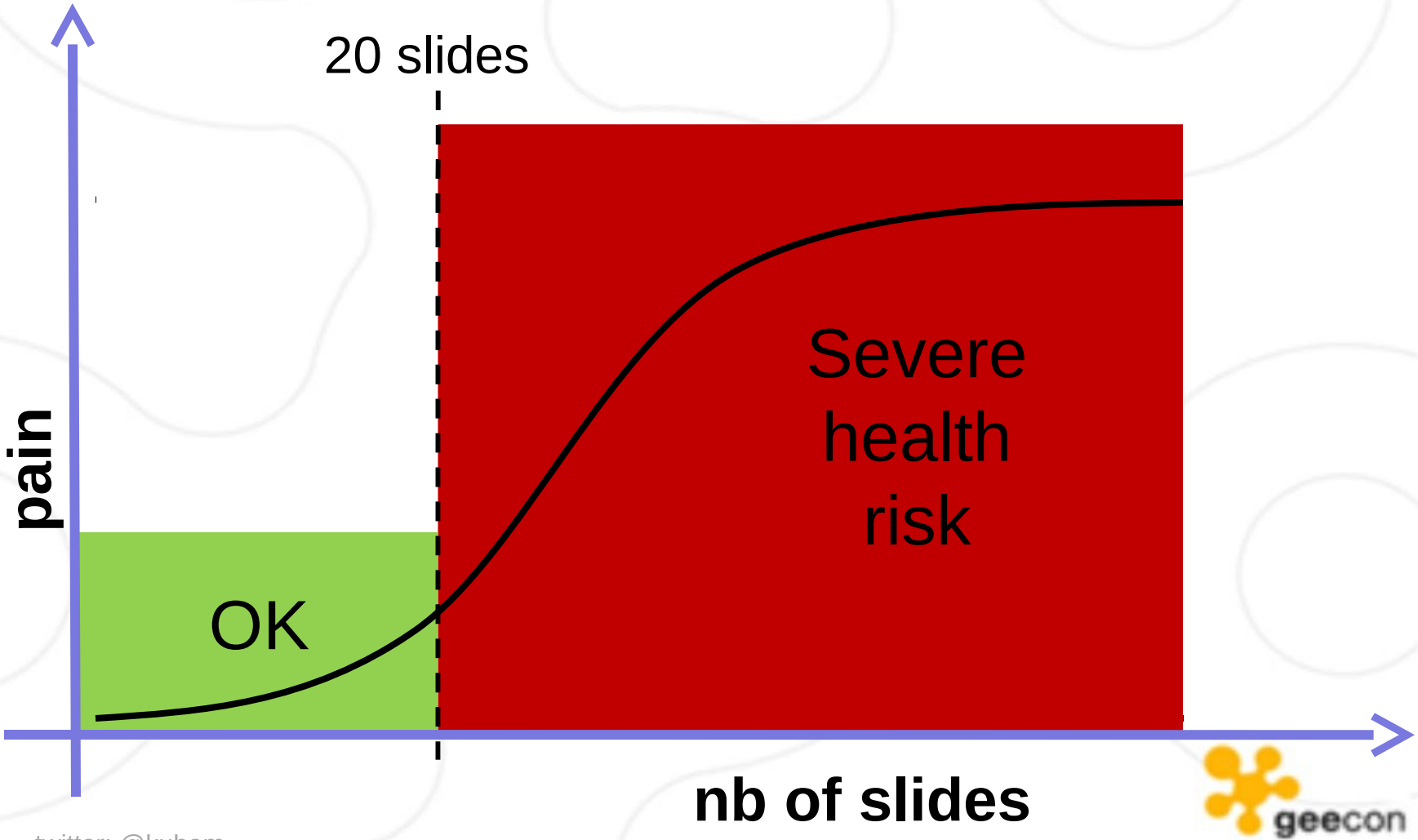
geecon

The experiment

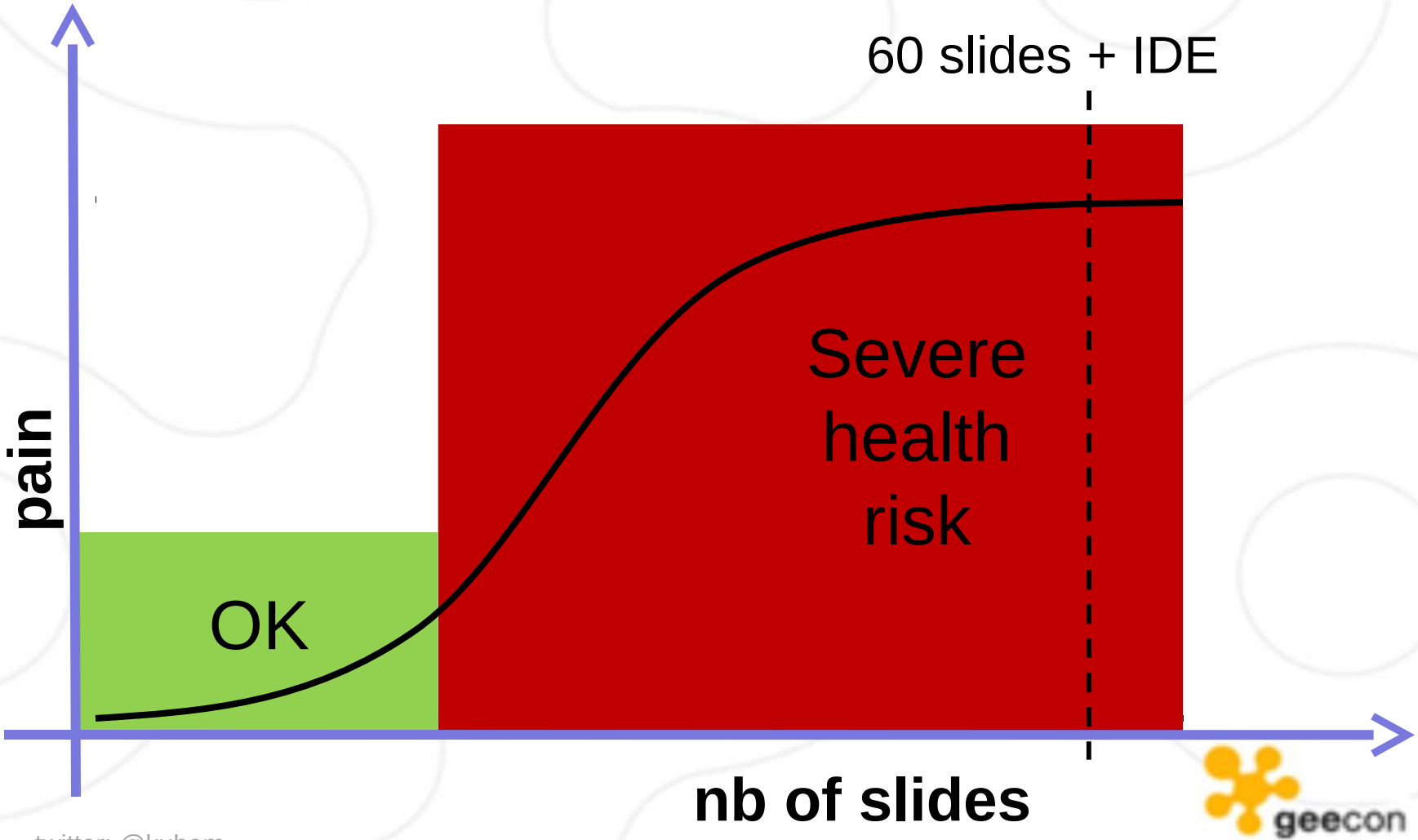


geecon

The experiment



The experiment



It started a year ago

GeeCon 2012 @ Poznań

twitter: @kubem

geecon

GeeCon 2012 @ Poznań

geecon

But in fact it started much earlier

JBoss 4.x

Taking form 7 to 20 minutes to start

No CI environment

geecon

Delivery a feature with only a single deploy

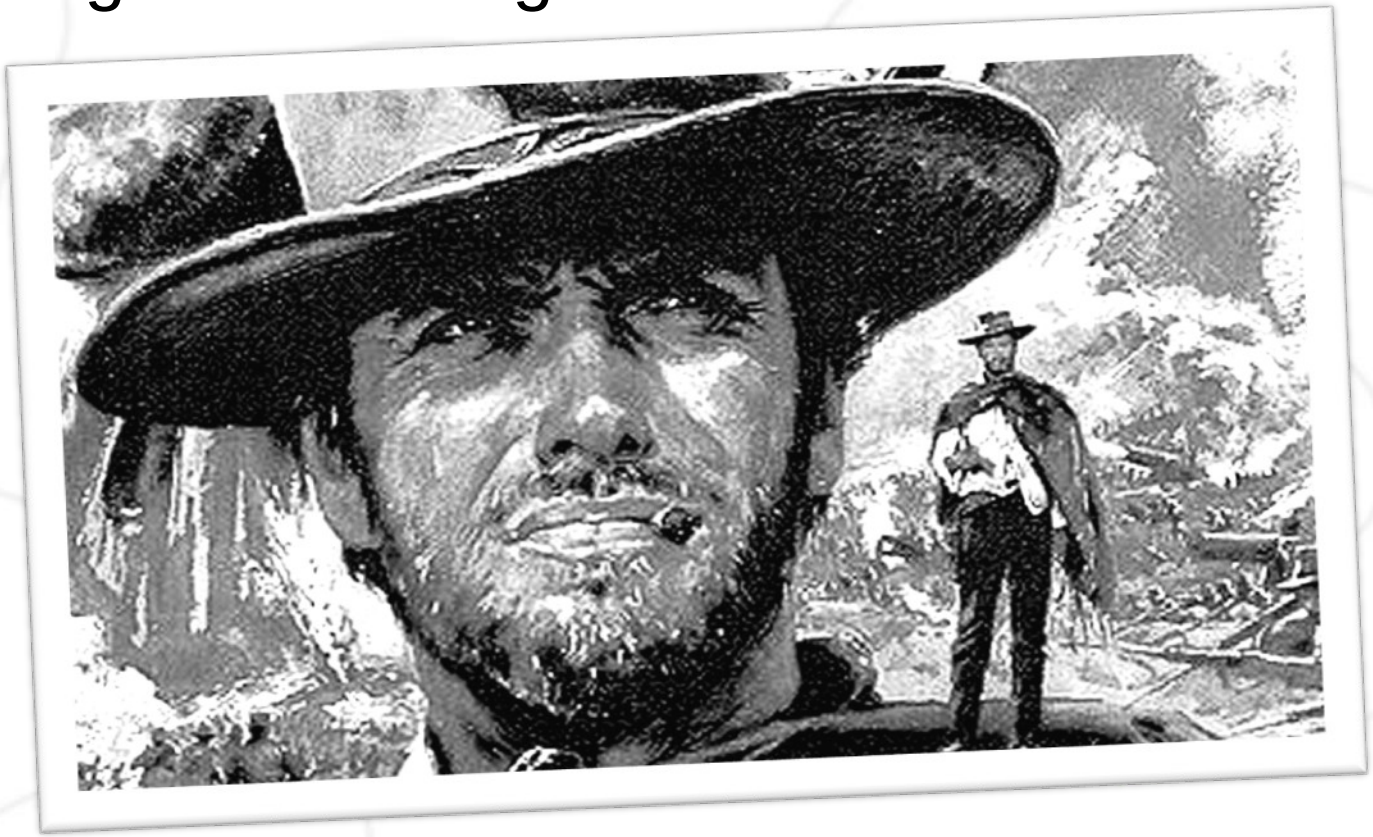
geecon

The Good

Embedded container

Your test is you component

Focused integration testing



```
INFO - Configuring Service(id=Default Security Service,
type=SecurityService, provider-id=Default Security Service)
INFO - Configuring Service(id=Default Transaction Manager,
type=TransactionManager, provider-id=Default Transaction Manager)
INFO - Creating TransactionManager(id=Default Transaction Manager)
INFO - Creating SecurityService(id=Default Security Service)
INFO - Configuring enterprise application:
/home/kubam/workspaces/java/various/jee-examples/stockapp/stockapp-ejb-
openejbtest/DozerQuotationConverterIntegrationTest
INFO - Auto-deploying ejb DozerConverterCache: EjbDeployment(deployment-
id=DozerConverterCache)
INFO - Auto-deploying ejb DozerQuotationConverter: EjbDeployment(deployment-
id=DozerQuotationConverter)
INFO - Configuring Service(id=Default Managed Container, type=Container,
provider-id=Default ManagedContainer)
INFO - Auto-creating a container for bean
pl.marchwicki.jee.stockapp.ejb.DozerQuotationConverterIntegrationTest:
Container(type=MANAGED, id=Default Managed Container)
INFO - Creating Container(id=Default Managed Container)
```

```java
@RunWith(ApplicationComposer.class)
public class HelloWorldServiceIntegrationTest {

    @EJB
    HelloWorldServiceLocal helloWorldService;

    @Test
    public void shouldReturnHello() {
        assertEquals(helloWorldService.sayHello("Jakub"), "Hello!
Jakub");
    }

    //...

}
```

```java
@RunWith(ApplicationComposer.class)
public class AuditMessageReceiverTest {

    final String originalTestMessage = "Hello World!";

    //..

    @PersistenceContext
    EntityManager em;

    @Test
    public void processQueue() throws JMSException {
        List<AuditLog> list = em.createQuery("from AuditLog",
                    AuditLog.class).getResultList();
        assertNotNull(list);
        assertEquals(1, list.size());
        assertEquals(originalTestMessage, list.get(0).getMessage());
    }

    //..

}
```

```java
@RunWith(ApplicationComposer.class)
public class AuditMessageProcessorTest {

    @Resource(name = "ConnectionFactory", mappedName =
"ConnectionFactory")
    ConnectionFactory connectionFactory;

    @Resource(name = "queue/audit", mappedName = "queue/audit")
    Queue queue;

    @Test(timeout = 2000)
    public void processQueue() throws Exception {
        final Connection connection =
          connectionFactory.createConnection();
        connection.start();
        final Session session = connection.createSession(false,
          Session.AUTO_ACKNOWLEDGE);

        final MessageConsumer incoming =
          session.createConsumer(queue);
        Message receive = incoming.receive();

        assertThat(receive, new IsInstanceOf(TextMessage.class));
        assertEquals(originalTestMessage, ((TextMessage)
          receive).getText());
    }

    //...
}
```

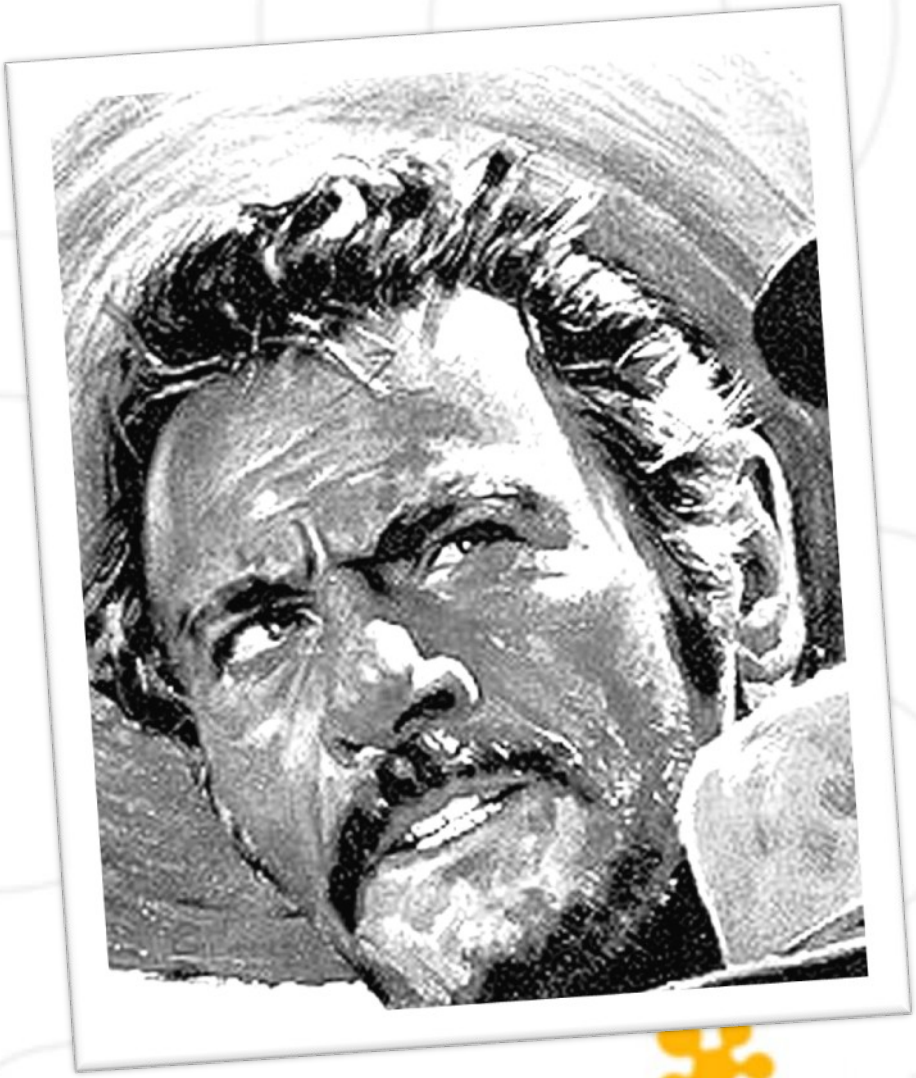
```java
@RunWith(ApplicationComposer.class)
public class HelloWorldServiceIntegrationTest {

    @Test
    public void shouldCallWebService() throws Exception {

        URL wsdlDocumentLocation =
            new URL("http://127.0.0.1:4204/module/HelloWorldService?
       wsdl");
        String namespaceURI =
       "http://hw.ejb.stockapp.jee.marchwicki.pl/";
        String servicePart = "HelloWorldServiceService";
        QName serviceQN = new QName(namespaceURI, servicePart);

        Service service = Service.create(wsdlDocumentLocation,
       serviceQN);
        HelloWorldServiceLocal helloWorld =
            service.getPort(HelloWorldServiceLocal.class);
        String s = helloWorld.sayHello("Jakub");

        assertEquals("Hello! Jakub", s);
    }

    //...

}
```

The Bad

Differences between containers

Vagueness of the EJB spec

No test with the actual server



JNDI tree

- {deploymentId} / {interfaceClass}
- {appName} / {moduleName} / {beanName}

But where it’s localted?

- java: **app/***
- java: **comp/env/***
- java: **global/***

Multiple names

```
public interface Service
{

    public String
hello();

}
```

Multiple names

```
public interface Service
{

    public String
hello();

}
```

```
@Stateless
public class MyService implements
Service {

    public String hello() {
       return "Hello world!";
    }

}
```

Multiple names

```
public interface Service
{

    public String
hello();

}
```

```
@Stateless
public class MyService implements
Service {

    public String hello()
       return "Hello worl
    }

}
```

```
@Stateless
public class MyOtherService implements
Service {

    public String hello() {
       return "Hello GeeCON!";
    }

}
```

Multiple names

```
public interface Service
{

    pub
hello()

}
```

```
@Startup
@Singleton
public class Client {

    @EJB
    Service service;

    @PostConstruct
    public void init() {
       System.out.println(service.hello());
    }

}
```

```
ts

}
```


geecon

```
DEPLOYMENTS IN ERROR:
  Deployment "vfs:///opt/java/jboss-
6.1.0.Final/server/default/deploy/module.jar" is in error due to the
following reason(s): java.lang.RuntimeException: Specified reference
[EJB Reference: beanInterface 'pl.marchwicki.jee6.Service', beanName
'null', mappedName 'null', lookupName 'null', owning unit
'ComponentDeploymentContext@2078856098{org.jboss.metadata.ejb.jboss.JBo
ssEnterpriseBeanMetaData.Client}'] was matched by more than one EJB:
[org.jboss.metadata.ejb.jboss.JBossSessionBean31MetaData@8b7ff749{MySer
vice},
org.jboss.metadata.ejb.jboss.JBossSessionBean31MetaData@5baed971{MyOthe
rService}].  Specify beanName explciitly or ensure beanInterface is
unique.
```

```
INFO: Created Ejb(deployment-id=Client, ejb-name=Client, container=My
Singleton Container)
May 11, 013 4:23:24 PM org.apache.openejb.assembler.classic.Assembler
startEjbs
INFO: Started Ejb(deployment-id=MyService, ejb-name=MyService,
container=My Stateless Container)
May 11, 2013 4:23:24 PM org.apache.openejb.assembler.classic.Assembler
startEjbs
INFO: Started Ejb(deployment-id=MyOtherService,
ejb-name=MyOtherService, container=My Stateless Container)
Hello world!
```

Different implementations

```
@XmlRootElement
public class Todo {

    private long id;

    @NotNull
    @Size(min = 1)
    private String title;

    private long order;

    private boolean completed;

    //...

}
```

Different implementations

```
@XmlRootElement
public class Todo {

    private long id;

    @NotNull
    @Size(min = 1)
    private String title;

    private long order;

    private boolean completed

    //...

}
```

```
@Path("/todos")
@Produces(MediaType.APPLICATION_JSON)
public class TodoResource {

    @GET
    public List<Todo> getAll() {
       return store.getAll();
    }

    @GET
    @Path("/{id}")
    public Todo get(@PathParam("id") long
       id) {
       return store.get(id);
    }

    //...

}
```

```
GET /todos/1 HTTP/1.1
Host: localhost

HTTP/1.1 200 OK
{

    "id": 1,
    "title": "First todo",
    "order": 1,
    "completed": false

}
```

```
GET /todos HTTP/1.1
Host: localhost

HTTP/1.1 200 OK
[

    {
        "id": 1,
        "title": "First todo",
        "order": 1,
        "completed": false
    },
    {
        "id": 2,
        "title": "Second todo",
        "order": 2,
        "completed": false
    }

]
```

```
GET /todos/1 HTTP/1.1
Host: localhost

HTTP/1.1 200 OK
{
    todo: {
        "id": 1,
        "title": "First todo",
        "order": 1,
        "completed": false
    }
}
```

```
GET /todos HTTP/1.1
Host: localhost

HTTP/1.1 200 OK
{
    todo: [
        {
            "id": 1,
            "title": "First todo",
            "order": 1,
            "completed": false
        },
        {
            "id": 2,
            "title": "Second todo",
            "order": 2,
            "completed": false
        }
    ]
}
```

```
GET /todos/1 HTTP/1.1
Host: localhost

HTTP/1.1 200 OK
{
    todo: {
        "id": 1,
        "title": "First todo",
        "order": 1,
        "completed": false
    }
}
```

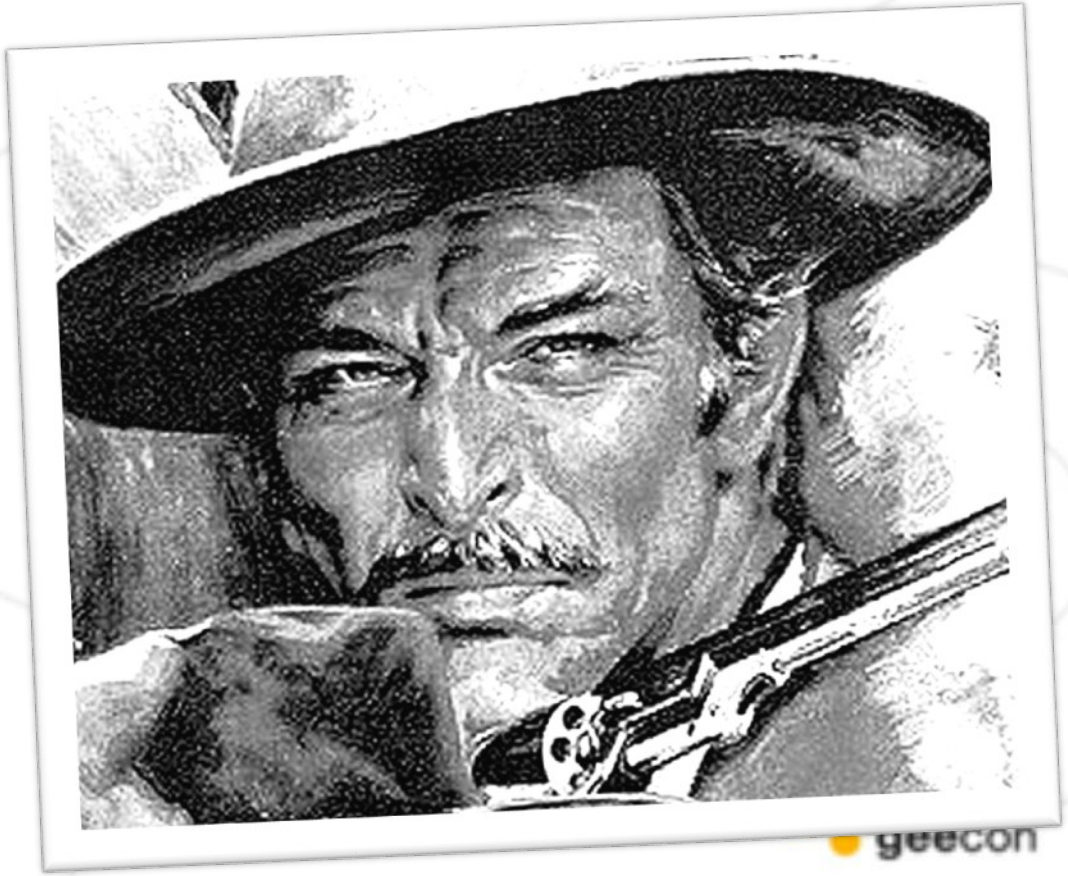
```
GET /todos HTTP/1.1
Host: localhost

HTTP/1.1 200 OK
{
    todo: [
        {
            "id": 1,
            "title": "First todo",
            "order": 1,
            "completed": false
        },
        {
            "id": 2,
            "title": "Second todo",
            "order": 2,
            "completed": false
        }
    ]
}
```

The Ugly

Loads of boilerplate code

Tests only EJBs

Never deployed



Configuration

```
@RunWith(ApplicationComposer.class)
public class AuditMessageReceiverTest {

    final String originalTestMessage = "Hello World!";

    //..

    @PersistenceContext
    EntityManager em;

    @Test
    public void processQueue() throws JMSException {
        List<AuditLog> list = em.createQuery("from AuditLog",
                    AuditLog.class).getResultList();
        assertNotNull(list);
        assertEquals(1, list.size());
        assertEquals(originalTestMessage, list.get(0).getMessage());
    }

    //..

}
```

```java
@RunWith(ApplicationComposer.class)
public class AuditMessageReceiverTest {

    @Module
    public EjbJar module() {
        final EjbJar ejbJar = new EjbJar();
        ejbJar.addEnterpriseBean(new
          StatelessBean(AuditMessageProcessing.class));
        ejbJar.addEnterpriseBean(new
          StatelessBean(AuditLoggerService.class));
        ejbJar.addEnterpriseBean(new
          MessageDrivenBean(AuditingMessageListerner.class));

        return ejbJar;
    }

    //..
```

```java
@RunWith(ApplicationComposer.class)
public class AuditMessageReceiverTest {

    //..

    @Module
    public PersistenceUnit persistence() {
        PersistenceUnit unit = new PersistenceUnit("stockapp-audit");
        unit.setJtaDataSource("DefaultDS");
        unit.setProvider(HibernatePersistence.class);
        unit.addClass(AuditLog.class);

        Properties dbProperties = new Properties();
        dbProperties.setProperty("hibernate.hbm2ddl.auto", "create-
          drop");
        dbProperties.setProperty("hibernate.dialect",
          "org.hibernate.dialect.HSQLDialect");
        dbProperties.setProperty"hibernate.use_sql_comments", "true");
        dbProperties.setProperty("hibernate.show_sql", "true");
        dbProperties.setProperty("hibernate.format_sql", "true");

        unit.setProperties(dbProperties);
        return unit;
    }

    //..
```

```java
@RunWith(ApplicationComposer.class)
public class AuditMessageReceiverTest {

    //..

    @Configuration
    public Properties config() throws Exception {
        Properties p = new Properties();
        p.put("DefaultDS", "new://Resource?type=DataSource");
        p.put("DefaultDS.JdbcDriver", "org.hsqldb.jdbcDriver");
        p.put("DefaultDS.JdbcUrl", "jdbc:hsqldb:mem:testdb");
        return p;
    }

    //..
```

Test only EJB

It’s not a real application server

Webservices: Apache CXF

JMS: Apache MQ

JPA: OpenJPA
Hibernate
(any provider)

Where we are now?

The aliens have landed

- Arquillian is a testing framework
- There are runners…
- Containers…
- & Enrichers

The aliens have landed

- Arquillian is a testing framework
- There are runners…
  - JUnit
  - TestNG
- Containers…
- & Enrichers

The aliens have landed

- Arquillian is a testing framework
- There are runners…
- Containers…
  - Weld
  - Tomcat
  - JBoss
  - Glassfish
- & Enrichers

The aliens have landed

- Arquillian is a testing framework
- There are runners…
- Containers…
- & Enrichers
  - glue layer between your code and the container: inject, track results

A natural evolution

geecon

The are three modes

- Embedded
  - You run your app server together with your code, in a single JVM
- Managed
  - Arquillian starts and stops your nodes, together with an app, in separate JVMs
- Remote
  - Everything is remotly deployed

And four ways for communicating results

- Local
  - Same JVM, local bindings
- Servlet 2.5
  - Pre JEE-6 containers
- Servlet 3.0
  - Leveraging async servlets
- JMX
  - Jboss AS 7 way

How it works in practice

Why we do it?

- Real test with real services (no mocks movement)
- With real environment
- Short feedback cycle

geecon

What’s next

- Persitence Extension (DBUnit)
- GWT Extension
- JSFUnit
- BPEL testing
- Performance
- … ?

Call for arms
Arquillian Test Porting Initiative

David Blevins
@dblevins

Follow

Need an army of volunteers to port tests to #Arquillian for #TomEE. Literally thousands of tests to port from all the ASF projects

5:36 AM - 21 Jan 13

8 RETWEETS 1 FAVORITE

geecon

https://issues.apache.org/jira/browse/TOMEE-746

http://tomee.apache.org/dev/arquillian-test-porting-initiative.html

Projects to tackle

- Active MQ
- Apache CXF
- TomEE & OpenEJB
- MyFaces
- OpenWenBeans
- BVal

Lemme show you

You can innovate on nonfunctional areas, such as devops, logging, metrics, etc, if your core business needs to be more conservative.

Innovation comes not just from using new technologies, but also from using new techniques with old boring technologies

Questions

https://github.com/kubamarchwicki/jug-jee6

http://goo.gl/Lod46